

Searching for a superlinear lower bounds for the Maximum Consecutive Subsums Problem and the $(\min, +)$ -convolution

Wilfredo Bardales Roncalla
PUC-RIO, Brazil
laber@inf.puc-rio.br

Eduardo Laber
PUC-RIO, Brazil
laber@inf.puc-rio.br

Ferdinando Cicalese
Department of Computer Science, University of Verona, Italy
ferdinando.cicalese@univr.it

September 21, 2015

Abstract

Given a sequence of n numbers, the MAXIMUM CONSECUTIVE SUBSUMS PROBLEM(MCSP) asks for the maximum consecutive sum of lengths ℓ for each $\ell = 1, \dots, n$. No algorithm is known for this problem which is significantly better than the naive quadratic solution. Nor a super linear lower bound is known. The best known bound for the MCSP is based on the computation of the $(\min, +)$ -convolution, another problem for which neither an $O(n^{2-\epsilon})$ upper bound is known nor a super linear lower bound. We show that the two problems are in fact computationally equivalent by providing linear reductions between them. Then, we concentrate on the problem of finding super linear lower bounds and provide empirical evidence for an $\Omega(n \log n)$ lower bounds for both problems in the decision tree model.

1 Introduction

Given a sequence $A = (a_1, a_2, \dots, a_n)$ of n numbers, the MCSP asks to compute the sequence m_1, \dots, m_n , where $m_\ell = \max_{i=1, \dots, n-\ell+1} \{a_i + \dots + a_{i+\ell-1}\}$, is the maximum over all consecutive subsums of length ℓ .

The MCSP appears in several scenarios of both theoretical and practical interest like approximate pattern matching [6], mass spectrometry data analysis [11], and in the problem of locating large empty regions in data sets [3]. Most work has been done for the case where the input sequence is binary, since in this case the MCSP coincides with the problem of constructing membership query indexes for jumbled pattern matching [6, 1, 14].

It is not difficult to come up with simple $O(n^2)$ solutions for the MCSP since each value m_ℓ can be easily computed in linear time by one pass over the input sequence. Surprisingly, despite the growing interest generated by this problem (see, e.g., [6, 9, 10, 18], and references therein quoted), no solution is known with running time $O(n^{2-\epsilon})$ for some constant $\epsilon > 0$, nor a lower bound better than the trivial $\Omega(n)$ is known.

Algorithms that produce approximate solutions for the MCSP are also known [10]. However, for the general case, the best available algorithm in the real RAM model runs in $O(n^2 / \log n)$ [6, 18] and makes use of the algorithm proposed in [4] for computing a $(\min, +)$ -convolution.

The $(\min, +)$ -convolution problem is a natural variation of the classical convolution problem: Given two sequences $X = (x_0, x_1, \dots, x_n)$ and $Y = (y_0, y_1, \dots, y_n)$ of real numbers, the $(\min, +)$ -convolution of X and Y is the sequence $z = z_0, z_1 \dots z_{2n}$, with $z_k = \min_{i=0, \dots, k} \{x_i + y_{k-i}\}$, for $k = 0, \dots, 2n$.

This problem has important applications in a variety of areas, including signal processing, pattern recognition, computer vision, and mathematical programming. According to [4], this problem has appeared frequently in the literature since Bellman's early work on dynamic programming.

Like for the MCSP, no strongly subquadratic algorithm appears to be known to compute the $(\min, +)$ -convolution. The best known algorithm for computing $(\min, +)$ -convolution runs in $O(n^2/\log n)$. Recently, Williams [19] has provided a Monte Carlo algorithm that computes the $(\min, +)$ convolution in $O(n^2/2^{\Omega(\log^{1/2} n)})$ time on the real RAM. Although, this bound is better than $O(n^2/\text{polylog}(n))$ it is still $\omega(n^{2-\epsilon})$ for any $\epsilon > 0$. For the special case of monotone increasing sequences with elements bounded by $O(n)$ a recent breakthrough in [7] shows that $(\min, +)$ -convolution can be computed in time $\tilde{O}(n^{1.864})$. This result implies an equivalent sub quadratic bound for MCSP on 0/1 sequences.

Taking into account the apparent difficulty to devise an $O(n^{2-\epsilon})$ algorithm for the MCSP and for computing the $(\min, +)$ -convolution, a natural question to ask is whether there exists a non-trivial superlinear lower bound for these problems.

The decision tree model is a widely used model of computation to study lower bounds for algorithmic problems. Fundamental algorithmic problems as searching, sorting and selection are examples of problems that have been studied in this model. In the decision tree model each algorithm can be represented by a decision tree, where internal nodes correspond to computations; the branches(edges) that leave an internal node v correspond to the possible results of the computation associated with v and the leaves correspond to the possible outputs of the algorithm (see section 3 for the exact definitions). Some variants/specializations of the decision tree model as linear decision trees and algebraic decision trees have also been used to study the complexity of algorithmic problems [2]. In [4] it is shown that the $(\min, +)$ -convolution of two n -element vectors can be computed in $O(n^{3/2})$ in the non-uniform linear decision tree model.

This work describes our findings in the quest for a super-linear lower bound for the MCSP in the decision tree model of computation.

Our Contributions. We provide computational evidences that the running time of both MCSP and $(\min, +)$ -convolution problem is $\Omega(n \log n)$ in the decision tree model of computation.

We first establish a linear equivalence between both problems by showing a linear reduction from $(\min, +)$ -convolution to MCSP. The opposite direction was shown in [3]. As a result of this equivalence, any bound for one problem also holds for the other. Then, in the following, we only concentrate on the MCSP.

In Section 3, we argue that a lower bound for the MCSP in the decision tree model can be obtained by generating a large set of inputs sequences such that no pair of them produces a common output. We present a construction which shows that there exists such a set of exponential size, although, this is still not enough to prove a superlinear lower bound on MCSP:

In Section 4, by using a deterministic approach we show empirically that for $n \leq 14$ there exists a set of inputs, with the above property, and cardinality larger than $(n/2)!$ so that $n/2 \log(n/2)$ is a lower bound on the depth of any decision tree that solves the MCSP

for instances of size $n \leq 14$. This required 27 hours of CPU time in our computational environment. In order to address larger values of n , we employed sampling strategies. We devised a hypothesis test and showed that the $n/2 \log(n/2)$ lower bound also holds for any $n \leq 100$ with confidence much larger than 99.999%.

Our results can bring new insight on the complexity of both the MCSP and the $(\min, +)$ -convolution problem and may represent a initial step towards proving a superlinear lower bound for these problems. Moreover, the techniques employed might be useful in the investigation of lower bounds for other computational problems with the same flavour.

2 An Equivalence between MCSP and the $(\min, +)$ -Convolution Problem

We start by showing that MCSP and $(\min, +)$ -convolution are computationally equivalent.

Theorem 1. *There exist linear reductions between the MCSP and the $(\min, +)$ -convolution.*

Proof. The reduction from MCSP to $(\min, +)$ -convolution is observed in [3] where the latter problem is presented under the geometric naming Lowest Midpoints Problem. Thus, we just present the reduction from the $(\min, +)$ -convolution to MCSP.

Let $I = (X, Y)$ be an input for the $(\min, +)$ -convolution, where $X = (x_0, \dots, x_n)$ and $Y = (y_0, \dots, y_n)$. Let S be a large enough number and define the input sequence $A = (a_1, \dots, a_{2n+4})$ for the MCSP as follows: $a_{n+1} = a_{n+4} = S$; $a_{n+3} = -y_0$; $a_{n+2} = -x_0$; for each $i < n + 1$ set $a_i = x_{n-i} - x_{n+1-i}$ and for $i > n + 4$ set $a_i = y_{i-n-5} - y_{i-n-4}$:

$$\begin{array}{cccccccccccccccc} a_1 & & \cdots & & a_n & & a_{n+1} & & a_{n+2} & & a_{n+3} & & a_{n+4} & & a_{n+5} & & a_{n+6} & & \cdots & & a_{2n+4} \\ (x_{n-1} - x_n) & & \cdots & & (x_0 - x_1) & & S & & -x_0 & & -y_0 & & S & & (y_0 - y_1) & & (y_1 - y_2) & & \cdots & & (y_{n-1} - y_n) \end{array}$$

Assuming $S > \sum_{i=0}^n |x_i| + |y_i|$, for any $k \geq 4$ a maximum sum of k consecutive elements includes a_{n+1} and a_{n+4} . Then, for each $k \geq 4$,

$$\begin{aligned} \max_{j=1, \dots, (2n+4)-k+1} \sum_{i=j}^{j+k-1} a_i &= \max_{\substack{0 \leq s, t \leq k-4 \\ s+t=k-4}} \sum_{i=n+1-t}^{n+4+s} a_i = \max_{\substack{0 \leq s', t' \leq k-4 \\ s'+t'=k-4}} 2S - x_{t'} - y_{s'} = \\ &= 2S - \min_{0 \leq t' \leq k-4} (x_{t'} + y_{k-4-t'}) = 2S - z_{k-4}. \end{aligned}$$

where, as above, z_0, \dots, z_{2n} denotes the result of the convolution of X and Y . Therefore, for each $k = 0, \dots, 2n$, we have the equivalence $z_k = 2S - \left(\sum_{j=p_k+4}^{p_k+4+k+3} a_j \right)$, where p_k is the starting position of a maximum consecutive sum of length k in A . \square

Due to these linear time reductions we can conclude that MCSP and $(\min, +)$ -convolution have the same time complexity. In the following, we will focus our discussion on lower bounds for MCSP and any conclusion reached will also hold for the $(\min, +)$ -convolution.

3 Towards a Lower Bound for the MCSP

In this section we discuss our approach to prove a lower bound for the MCSP. It will be convenient to employ the following alternative formulation of the MCSP

Input. A sequence $A = (a_1, \dots, a_n)$ of n real numbers;

Output. A sequence $P = (p_1, \dots, p_n)$, where p_ℓ , for $\ell = 1, \dots, n$, is the starting position of a consecutive subsequence of A that has maximum sum among the consecutive subsequences of A with length ℓ , i.e., the subsequence $a_{p_\ell} a_{p_\ell+1} \dots a_{p_\ell+\ell-1}$ is a maximum consecutive subsum of length ℓ .

We call the sequence P an *output configuration* or simply a *configuration*.

For example, for the input sequence $A = (3, 0, 5, 0, 2, 4)$ the only output configuration is given by the sequence $P = (5, 5, 1, 3, 2, 1)$, which says, e.g., that there is a maximum consecutive subsum of length 4 starting at position 3.

We note that there are $n!$ possible configurations because p_i , for $i = 1, \dots, n$, can assume any value in the set $\{1, \dots, n - i + 1\}$. In particular for the input $A = (a_1, \dots, a_n)$, with $a_1 = a_2 = \dots = a_n = 1$, all the $n!$ possible configurations are output configurations for A . This example also shows that some input sequences have more than one output configuration.

We will study the above version of MCSP in the *decision tree model*. An algorithm in this model is a ternary tree. Each internal node I contains a test $f(I) \leq 0$? for some rational function f of n (size of the input) arguments¹. Each leaf of the tree contains a set of function g_i ($i = 1, \dots, n$) on the n -valued input. For any input $A = (a_1, \dots, a_n)$, the algorithm moves from the root down the tree. At each node the corresponding test is performed and a branch is followed according to whether the test outcome is > 0 , < 0 , $= 0$. When a leaf is reached, the output $P = (p_1, \dots, p_n)$ is given by $p_i = g_i(A)$. The cost of the algorithm is defined to be the height of the tree. The complexity $K(n)$ in this model is the minimum cost of any such algorithm. We will be concerned with an information theoretic lower bound on $K(n)$.

3.1 An approach based on unique configurations

For any input A for the MCSP we define $\mathcal{P}(A) = \{P \mid P \text{ is an output configuration for } A\}$.

We say that a configuration P is *unique* if and only if there exists an input A for the MCSP for which $\mathcal{P}(A) = \{P\}$.

Our approach to prove a lower bound on MCSP consists on finding a large set of distinct unique configurations.

In fact, let $\mathbb{P} = \{P_1, P_2, \dots, P_k\}$ be a set of distinct unique configurations. Moreover, let A_i , for $i = 1, \dots, k$, be an instance for MCSP such that P_i is its unique configuration, i.e., $\mathcal{P}(A_i) = \{P_i\}$. Then, the instances in the set $\mathbb{A} = \{A_1, \dots, A_k\}$ are distinct. Hence, in any decision tree, for any distinct inputs A_i, A_j , the leaves associated to the corresponding outputs must be distinct, hence the decision tree must have at least k leaves. Then, the height of the tree is at least $\lceil \log_3 k \rceil$, which shows that $\lceil \log_3 k \rceil$ is a lower bound to the problem in the decision tree model. We have proved the following.

Theorem 2. *If \mathbb{P} is a subset of distinct unique configurations, then $K(n) = \Omega(\log |\mathbb{P}|)$.*

We shall observe that if every configuration were unique, then we could prove a lower bound of $\Omega(n \log n)$ since there exists $n!$ configurations of size n . However, there are configurations that are not unique like the configuration $P = (1, 2, 1)$. In fact, assume that $A = (a_1, a_2, a_3)$ is an input for which P is its unique configuration. Then, we would have both $a_1 > a_3$ and $a_2 + a_3 > a_1 + a_2$, which is not possible.

The following construction shows that the number of unique configurations is indeed very large.

¹If all functions are restricted to be linear (resp. polynomials) the model is referred to as the *linear* (resp. algebraic) decision tree model.

Theorem 3. *There exist $\Omega(2^n)$ unique configurations.*

Proof. Fix a subset $S \subseteq \{4, 5, \dots, n\}$ and define

$$a_i^S = \begin{cases} 0 & \text{if } i = 1 \\ 2 & \text{if } i = 2 \\ 4n & \text{if } i = 3 \\ 3 & \text{if } i \geq 4 \text{ and } i \notin S \\ 1 & \text{if } i \geq 4 \text{ and } i \in S \end{cases} \quad p_j^S = \begin{cases} 3 & \text{if } j \leq n-2 \text{ and } (j+2) \notin S \\ 2 & \text{if } j \leq n-2 \text{ and } (j+2) \in S \\ n-j+1 & \text{if } j \geq n-1 \end{cases}$$

We first show that for the input $A^S = (a_1^S, \dots, a_n^S)$ the unique output configuration is $P^S = (p_1^S, \dots, p_n^S)$. For this we need to verify that for each $j = 1, \dots, n$, in A there is only one maximum consecutive subsum of length j and it starts at p_j^S as given above.

The statement is trivially true for $j = n$ and for $j = n-1$ since in the latter case it is enough to observe that $0 = a_1 < a_2$.

Since $a_3 = 4n > \sum_{i \neq 3} a_i$ it follows easily that for any $1 \leq k \leq n-2$ a maximum consecutive sum of length k must contain the element a_3 . Any such sum will then start at one of the first 3 elements. However, since any element other than a_1 is greater than zero, it also follows that a maximum consecutive sum of size $k \in \{1, \dots, n-2\}$ must start at a_2 or a_3 . In formula, for $1 \leq k \leq n-2$, let s_k denote the maximum consecutive sum of size k , then $s_k \in \{\sum_{i=2}^{k+1} a_i, \sum_{i=3}^{k+2} a_i\}$. Let Δ_k be the difference between the two possible values for s_k . We have $\Delta_k = \sum_{i=2}^{k+1} a_i - \sum_{i=3}^{k+2} a_i = a_2 - a_{k+2} = 2 - a_{k+2}$. Thus, $\Delta_k > 0$ if $k \in S$ and $\Delta_k < 0$ otherwise, that is, $s_k = \sum_{i=2}^{k+1} a_i$ if $k \in S$ and $s_k = \sum_{i=3}^{k+2} a_i$ if $k \notin S$, which proves the above claim. The uniqueness of the configuration follows from the fact that all the inequalities in the above arguments are strict.

Therefore, for each $S \subseteq \{4, \dots, n\}$ we obtain an unique output configuration. Since there are $2^{n/8}$ distinct subsets of $\{4, \dots, n\}$ and each of them corresponds to a distinct unique configuration we have the desired result. □

The existence of at least exponentially many (in n) configurations supports our approach and is an indication of its potential. However, this result combined with Theorem 2 is still not enough to obtain a non trivial (superlinear) lower bound for the MCSP. This motivated us to enrich our analysis by empirically exploring the number of unique configurations.

4 Empirical evidences that MCSP requires $\Omega(n \log n)$ time

In order to count the number of unique configurations it is important to decide whether a given configuration P is unique or not. For that we test whether there exists an input sequence $A = (a_1, \dots, a_n)$ for which P is its unique output configuration. For $i \in \{1, \dots, n-1\}$ and a configuration $P = (p_1, \dots, p_n)$, let $\mathcal{Q}(P, i)$ be the following set of inequalities:

$$\sum_{k=p_i}^{p_i+i-1} a_k > \sum_{k=j}^{j+i-1} a_k \quad \text{for } j = 1, \dots, n-i+1 \text{ and } j \neq p_i$$

It is easy to realize that the contiguous subsequence of length i that starts at position p_i of A has sum larger than the sum of any other contiguous subsequence of length i if and only if the point $A = (a_1, \dots, a_n) \in R^n$ satisfies the above set of inequalities. Thus, P is a unique configuration if and only if the set of inequalities $\mathcal{Q}(P, 1) \cup \mathcal{Q}(P, 2) \cup \dots \cup \mathcal{Q}(P, n-1)$ has a feasible solution. In our experiments we employed a linear programming solver to perform this verification.

In order to speed up our computation we also employ a sufficient condition for the non-uniqueness of a configuration, which is provided by the following proposition.

Proposition 1 (Non-Adjacency Property). *Let $P = (p_1, \dots, p_n)$ be an output configuration. If there exist $i, j \in \{1, \dots, n\}$ such that $p_j = p_i + i$, then P is not unique.*

Proof. Let i, j be such that $p_j = p_i + i$. In addition, assume that there is an input A for which P is its unique configuration. This means that for any $r \neq i$ and $u \neq j$ we have

$$\sum_{s=r}^{r+i-1} a_s < \sum_{s=p_i}^{p_i+i-1} a_s \quad \sum_{s=u}^{u+j-1} a_s < \sum_{s=p_j}^{p_j+j-1} a_s \quad (1)$$

Let $m = \min\{i, j\}$ and let's split the sequence $a_{p_i} a_{p_i+1} \dots a_{p_i+i-1} a_{p_j} a_{p_j+1} \dots a_{p_j+j-1}$ into three parts, and let R_1, R_2, R_3 be the intervals of indices of the elements in these three parts defined as follows:

$$R_1 = \{p_i, p_i+1, \dots, p_i+m-1\}, R_2 = \{p_i+m, p_i+m+1, \dots, p_j+j-1-m\}, R_3 = \{p_j+j-m, \dots, p_j+j-1\}.$$

Case 1. If $i < j$ we have $\sum_{s=p_i}^{p_i+j-1} a_s = \sum_{s \in R_1} a_s + \sum_{s \in R_2} a_s$ and $\sum_{s=p_j}^{p_j+j-1} a_s = \sum_{s \in R_2} a_s + \sum_{s \in R_3} a_s$. Then applying the second inequality in (1) we get

$$\sum_{s \in R_1} a_s + \sum_{s \in R_2} a_s = \sum_{s=p_i}^{p_i+j-1} a_s < \sum_{s=p_j}^{p_j+j-1} a_s = \sum_{s \in R_2} a_s + \sum_{s \in R_3} a_s,$$

which implies that $\sum_{s=p_i}^{p_i+i-1} a_s = \sum_{s \in R_1} a_s < \sum_{s \in R_3} a_s = \sum_{s=p_j+j-i}^{p_j+j-1} a_s$ which is a contradiction to the first inequality in (1).

Case 2. If $i > j$ we have $\sum_{s=p_i}^{p_i+i-1} a_s = \sum_{s \in R_1} a_s + \sum_{s \in R_2} a_s$ and $\sum_{s=p_i+j}^{p_i+j-1} a_s = \sum_{s \in R_2} a_s + \sum_{s \in R_3} a_s$. Then applying the first inequality in (1) we get

$$\sum_{s \in R_1} a_s + \sum_{s \in R_2} a_s = \sum_{s=p_i}^{p_i+i-1} a_s > \sum_{s=p_i+j}^{p_i+j-1} a_s = \sum_{s \in R_2} a_s + \sum_{s \in R_3} a_s,$$

which implies that $\sum_{s=p_i}^{p_i+j-1} a_s = \sum_{s \in R_1} a_s > \sum_{s \in R_3} a_s = \sum_{s \in p_j}^{p_j+j-1} a_s$ which is a contradiction to the second inequality in (1).

Since in either case we have a contradiction, it follows that if i, j are adjacent maxima than P cannot be a unique configuration. \square

For instance, consider the configuration $P = (5, 1, 3, 4, 1)$. By taking $i = 2$ and $j = 3$, we have $p_i + i = p_j$. Thus, we can conclude that P is not unique, as can be easily verified.

Unfortunately, this non-adjacency property does not completely characterize the set of unique configurations because there exist configurations with no adjacent maximums that are not unique. For example, the configuration $P = (2, 4, 2, 1, 2, 1)$ has no adjacent maximums and is not unique. In fact, if $A = (a_1, \dots, a_6)$ is an input sequence for which P is its unique configuration then we must have simultaneously: (i) $a_2 > a_4$ because of $p_1 = 2$; (ii) $a_4 + a_5 > a_1 + a_2$ because of $p_2 = 4$; and (iii) $a_1 + a_2 + a_3 + a_4 > a_2 + a_3 + a_4 + a_5$ because of $p_4 = 1$. However, this is impossible, since by summing the first two inequality and adding a_3 on both sides, we obtain a contradiction to the third inequality. Nonetheless, we can use the above condition to speed up our algorithms.

In order to exactly count the number of unique configurations, we explore the tree of all permutations of $\{1, \dots, n\}$, pruning the nodes that do not lead to an unique configuration. In fact, assume that the values of the positions $1, 2, \dots, i-1$ of the permutation P , under construction, are already fixed. Then, for each j that does not appear in the first $i-1$ positions, the procedure $\text{ISFEASIBLE}(P, i, j)$, explained below, is called to verify whether it is possible to extend the partial permutation P by setting the value of position i to j . If this is the case, the algorithm set $p_i = j$ and it proceeds constructing the permutations. Whenever we complete a permutation we increase the number of unique configurations.

The procedure $\text{IS-FEASIBLE}(P, i, j)$ first verifies if the subsequence of A starting at position j is adjacent to some maximum subsequence that has already been fixed. If this test is positive it rules out j as a value for p_i due to Proposition 1. Otherwise, the procedure verifies whether the set of inequalities $Q(P, 1) \cup \dots \cup Q(P, i)$ is feasible and it returns TRUE or FALSE, accordingly.

Algorithm 1 $\text{ISFEASIBLE}(P, i, j)$

```

1: if the subsequence of length  $i$  starting at position  $j$  is adjacent to the subsequence of length  $k$ 
   starting at  $p_k$  for some  $k < i$  then
2:   return FALSE
3: else
4:    $p_i = j$     % this is only to verify if this extension if feasible.
5:   if the set of inequalities  $Q(P, 1) \cup \dots \cup Q(P, i)$  is feasible
6:     then return TRUE
7:   else return FALSE

```

Table 1 presents the results obtained by the deterministic approach. We were able to determine the number of unique configurations up to $n = 14$. The results suggest a super exponential growth. Indeed, notice the growth of the ratio between the number of unique configurations and $(n/2)!$. We selected the function $(n/2)!$ because it is a simple function whose logarithm is $\theta(n \log n)$.

All the executions required 27 hours of CPU time under the following hardware and software specifications: Main Hardware Platform: Intel® Core™ i7 3960X, 3.30GHz CPU, 32GB RAM, 64-bit; OS: Windows 7 Professional x64; Compiler: Microsoft® Visual C# 2010 Compiler version 4.0.30319.1; Solver: Gurobi Optimizer.

Table 1: The number of unique configurations for $n = 1, \dots, 14$ compared to the value $n/2!$.

n	$U(n) = N^{\circ}$	Unique Config.	$\frac{n}{2}!$	Ratio $\frac{U(n)}{(n/2)!}$
1		1	0.8	1.25x
2		2	1.0	2.00x
3		4	1.3	3.07x
4		12	2.0	6.00x
5		36	3.3	10.90x
6		148	6.0	24.66x
7		586	11.6	50.51x
8		2,790	24.0	116.25x
9		13,338	52.3	255.02x
10		71,562	120.0	596.35x
11		378,024	287.9	1,313.03x
12		2,222,536	720.0	3,086.85x
13		12,770,406	1,871.3	6,824.34x
14		78,968,306	5,040.0	15,668.31x

In order to extend our analysis to larger instances we then employed a probabilistic approach.

4.1 A Probabilistic Approach

The first idea for estimating the number of unique configurations is to sample a large number M of configurations and test whether each of them is unique or not. The number of unique configurations found over M is an unbiased estimator for the number of unique configurations. With this approach we managed to obtain strong evidence of the super linear lower bound for n up to 28. To extend our range we followed a different approach.

In the deterministic case, we explore the configuration space via a depth first search over the back-tracking tree of all possible configurations. In our probabilistic approach, presented in Algorithm 2, we randomly traverse a path in this tree that corresponds to a unique configuration. Assume that we have already fixed the values for the positions $1, 2, \dots, i-1$ of the configuration P that is under construction. Then, in order to set the value of p_i , we construct a list S of all values $j \in \{1, \dots, n-i+1\}$ such that $\text{IsFEASIBLE}(P, i, j)$ returns TRUE. Let $b_i = |S|$ be the *branching factor* of our path at level i . Then, we randomly choose one of the values in S for p_i and continue to set the values of p_j for $j > i$; if the method observes the branching factors b_1, b_2, \dots, b_n , in a root to leaf path, then it outputs $X = \prod_{i=1}^n b_i$ as a guess for the number of unique configurations.

The value X can be used to estimate the number of unique configurations because X is a sample of a random variable \mathbf{X} whose expected value $E[\mathbf{X}]$ is equal to the number of the unique configurations. In fact, let ℓ be a leaf located at depth n of the backtracking tree, that is, ℓ corresponds to a unique configuration. The probability of reaching ℓ in our random walk is $1/B(\ell)$, where $B(\ell)$ is the product of the branching factors in the path from the root of the tree to ℓ . In addition, if ℓ is reached, the method outputs $B(\ell)$. Let L be the set of leaves

Algorithm 2 BRANCHING-PRODUCT(n)

```
1:  $X \leftarrow 1$ 
2: for  $i \leftarrow 1$  to  $n$  do
3:    $S \leftarrow \emptyset$ 
4:   for  $j \leftarrow 1$  to  $n - i + 1$  do
5:     if ISFEASIBLE( $P, i, j$ ) then
6:       Add  $j$  to the list  $S$  of possible branchings
7:   if  $S$  is empty then
8:     return 0
9:   else
10:     $X \leftarrow X \times |S|$ 
11:     $p_i \leftarrow$  value randomly selected from list  $S$ 
12: return  $X$ 
```

located at level n in the backtracking tree. Thus we have that

$$\mathbb{E}[\mathbf{X}] = \sum_{\ell \in L} \frac{1}{B(\ell)} \times B(\ell) = |L|.$$

After coming up with this approach, we found out that it had already been proposed to study heuristics for backtracking algorithms [17, 16].

We do not use directly the observed value X to estimate the number of unique configurations; instead, we assume as a null hypothesis that $\mathbb{E}[\mathbf{X}]$, the number of unique configurations, is smaller than or equal to $(\frac{n}{2}!)$ and use the sampled value of \mathbf{X} to reject this hypothesis with some level of confidence.

Let $c \geq 1$. Under the hypothesis that $\mathbb{E}[\mathbf{X}] \leq (\frac{n}{2}!)$, using Markov's inequality, it follows that:

$$\Pr \left[\mathbf{X} \geq c \frac{n}{2}! \right] \leq \Pr[X \geq c \mathbb{E}[\mathbf{X}]] \leq \frac{1}{c}$$

which implies that $\Pr \left[\mathbf{X} < c \frac{n}{2}! \right] \geq 1 - \frac{1}{c}$.

Therefore, if we sample \mathbf{X} and find a value larger than $c \frac{n}{2}!$, we can reject the hypothesis and conclude that the number of unique configurations is larger than $\frac{n}{2}!$ with confidence of $1 - \frac{1}{c}$.

We can extend this approach by taking the maximum of k samples. Let X_1, \dots, X_k be the values for k samples of the random variable \mathbf{X} . Then, with the hypothesis $\mathbb{E}[\mathbf{X}] \leq \frac{n}{2}!$ and using the above inequality we have

$$\Pr \left[\max\{X_1, \dots, X_k\} < c \frac{n}{2}! \right] = \Pr \left[\bigwedge_{i=1}^k \left(X_i < c \frac{n}{2}! \right) \right] = \prod_{i=1}^k \Pr \left[X_i < c \frac{n}{2}! \right] \geq \left(1 - \frac{1}{c} \right)^k. \quad (2)$$

Thus, if one of the values X_1, X_2, \dots, X_k is greater than or equal to $c \frac{n}{2}!$, then we can reject the hypothesis and conclude that $\mathbb{E}[\mathbf{X}] \geq \frac{n}{2}!$ with confidence $(1 - 1/c)^k$.

In our experiments we sampled 1000 values of \mathbf{X} for different values of n . The choice of 1000 for k was to guarantee a reasonable CPU time. Let $\max(n, 1000)$ be the maximum value found in the 1,000 samples and let $c_n = \lfloor \max(n, 1000) / \frac{n}{2}! \rfloor$. Table 2 shows the probability of $\Pr[\max\{X_1, \dots, X_{1000}\} < c_n \frac{n}{2}!]$ assuming that $\mathbb{E}[\mathbf{X}] \leq \frac{n}{2}!$ for configurations up to size $n = 100$. This probability also expresses our confidence to reject the hypothesis because in fact we've found a value equal to $c_n \frac{n}{2}!$. Based on these results we state the following conjecture.

Table 2: $\Pr [\max\{X_1, \dots, X_k\} < c_n \frac{n!}{2}]$ for $k = 1,000$

n	c_n	$\Pr[\max_1^k\{X_i\} < c_n \frac{n!}{2}]$
10	6,048	99.98346560846560%
11	23,760	99.99579124579120%
12	38,880	99.99742798353910%
13	439,296	99.99977236305360%
14	558,835	99.99982105636870%
20	372,252,672	99.9999973136530%
30	102,827,922,078	99.9999999902750%
40	4,680,410,711,674	99.9999999997860%
50	69,590,788,615,385	99.9999999999860%
60	562,841,769,233,371	99.9999999999980%
70	136,904,322,455,757	99.9999999999930%
80	87,399,891,508,924	99.9999999999890%
90	73,279,283,017	99.9999999863540%
100	204,252,401	99.9999951040970%

Conjecture 1. *The running time of MCSP is $\Omega(n \log n)$ in the decision tree model.*

References

- [1] G. Badkobeh, G. Fici, S. Kroon, Z. Lipták, Binary jumbled string matching for highly run-length compressible texts, IPL 113:604-608, 2013.
- [2] Ben-Or, Lower bounds for algebraic computation trees, in: STOC: ACM Symposium on Theory of Computing (STOC), 1983.
- [3] A. Bergkvist, P. Damaschke, Fast algorithms for finding disjoint subsequences with extremal densities, Pattern Recognition 39, 2281-2292, 2006.
- [4] D. Bremner, T.M. Chan, E.D. Demaine, J. Erickson, F. Hurtado, J. Iacono, S. Langerman, M. Patrascu, P. Taslakian, Necklaces, convolutions, and X+Y. Algorithmica 69, 294-314, 2014.
- [5] P. Burcsi, F. Cicalese, G. Fici and Zs. Lipták, Algorithms for jumbled pattern matching in strings, IJFCS, 23, 357-374, 2012.
- [6] P. Burcsi, F. Cicalese, G. Fici and Zs. Lipták, On approximate jumbled pattern matching in strings, Th. of Comp. Systems 50(1), 35-51, 2012.
- [7] T. Chan, M. Lewenstein, Clustered Integer 3SUM via Additive Combinatorics, in Proc. of STOC 2015.
- [8] T. Chan, A. Amir, M. Lewenstein, N. Lewenstein, On hardness of jumbled indexing, in: ICALP, 2014.
- [9] F. Cicalese, G. Fici and Zs. Lipták, Searching for jumbled patterns in strings, Proc. of the Prague Stringology Conference, pp. 105-117, 2009.

- [10] F. Cicalese, E. Laber, O. Weimann and R. Yuster, Near linear time construction of an approximate index for all maximum consecutive sub-sums of a sequence, Proc. of CPM2012, LNCS 7354, pp. 149-158, 2012.
- [11] M. Cieliebak, T. Erlebach, Z. Lipták, J. Stoye and E. Welzl, Algorithmic complexity of protein identification: combinatorics of weighted strings, Discrete Applied Mathematics 137(1), 27-46, 2004.
- [12] D. Eppstein, Efficient algorithms for sequence analysis with concave and convex gap costs, PhD thesis, Comp. Sc. Dept., Columbia Univ., 1989.
- [13] T. Fukuda, Y. Morimoto, S. Morishita and T. Tokuyama, Data mining with optimized two-dimensional association rules, ACM Trans. Database Syst. 26, 179-213, 2001.
- [14] T. Gagie, D. Hermelin, G. M. Landau, O. Weimann, Binary Jumbled Pattern Matching on Trees and Tree-Like Structures, Proc. of ESA 2013, LNCS 8125, pp. 517-528, 2013
- [15] R. Karp, Reducibility among combinatorial problems, Complexity of Computer Computations, eds. R. Miller and J. Thatcher (Plenum Press, 1972), pp. 85-103.
- [16] D. E. Knuth, Estimating the efficiency of backtrack programs, Mathematics of computation 29(129), 122-136, 1975.
- [17] O. Kullmann, Fundaments of branching heuristics, Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications 185, pp. 205-244, 2009. Comput. Sci. 377 (May 2007) 151-156.
- [18] T.M. Moosa and M.S.Rahman, Sub-quadratic time and linear space data structures for permutation matching in binary strings, J. of Discrete Algorithms 10, pp. 5-9, 2012.
- [19] R. Williams, Faster all-pairs shortest paths via circuit complexity, in: STOC: ACM Symposium on Theory of Computing (STOC), 2014.